COMPAQ
Inspiration Technology

Getting the Most Out of Big Iron

Platform Design Using a Flexible Architecture:

Vendor and Customer Choices

(and some other digressions……..)

Ed Benson

COMPAQ

Bad timing…….

- HP Compaq merger is nearly official but….
  - Until deal closes we are still competitors
  - Information sharing is very restricted
    - I'm not privy to HP planning
  - When the deal closes, there will be new roadmaps
  - Hence I will avoid product specific details

Scale of the problem with BIG IRON

"It is not enough that the hardware capabilities scales; the entire system solution must scale, including the protocols used to realize programming models and the capabilities provided by the operating system, such as process scheduling, storage management, and I/O."

Sec 7.1.5 – Parallel Computer Architecture – Culler and Singh

Platform verses Environment

I find it hard to separate the “platform” from the entire “environment” in terms of getting the MOST out of BIG IRON. Please accept the scope creep of this talk.

Grounding….. I'm sure some will disagree

- Core CPU performance is not an issue
- Memory latency will remain a challenge
- Local memory bandwidth continues to improve
  - EV7 integral RAMBUS controller
- Remote memory latency via an inter-connect will not improve significantly compared to CPU improvement – speed of light, interface crossings
- Remote memory bandwidth ratios can be maintained
- Parallelism of systems must increase

COMPAQ

My view of the state of BIG IRON

- Coarse Architectural Convergence - Clusters
  - No research or market to indicate shift any time soon
- Implementation Divergence and Linux/Beowulf dilution
- Support/Infrastructure Weakness
  - Dollars/Market available for non-essentials
- Relative low volume – sites, developers
- Diverse missions – limited reuse – momentum
- Scalable parallel programming will remain hard

Architectural Convergence

- Standard SMP building blocks
- Low overhead inter-connect with communications assist
  - Stepped bandwidth increases at a rate less than Moore's Law
  - Multiple rails for performance and availability
- IO bridge/bus is port to memory for inter-connect
  - Limiting latency improvements
- Hence a deep and complex memory hierarchy
- Full UNIX capable environment

Implementation Divergence

- CPU chip and therefore SMP building block
- Inter-connect and topology
- UNIX OS – vendor specific control and performance calls
  - Over time Linux MAY change this
- I/O
  - Common parts but not architecture
  - Customers either love or hate local I/O
- Resource management
- System management

COMPAQ

Support/Infrastructure Weakness

- Few large scale Independent Software Vendor (ISV) supplied applications
- Middleware – e.g.,DCE/DFS
- Development tools
- Performance tools
- Skilled developers and users

COMPAQ

Investment?

Is the BIG IRON market big enough and profitable enough to support the current level of divergence and deliver performance?

Would more convergence allow for better focused investment? (How many resource management facilities do we need?)

Diverse Linux/Open Source efforts helping?

Are adequate dollars being spent to sponsor the development of AND purchase of tools?

COMPAQ

Getting the most WHAT out of BIG IRON?

- Peak chest-pounding paper based FLOPS?
- Top500 LINPACK? (just add memory ☺)
- Real application capability?
  - Weak and strong scaling?
- Sustained capacity?
- I/O?

- First two above – get too much attention !!
- As a vendor we have to worry about them all

Some Big Challenges to getting the most…

- Program design and programming skills
  - Parallel programming is hard
  - Expertise is hard to scale
- Overhead creep from integration
- Reliability/Availability – Failure handling
- Latency – inter-connect and elsewhere
- I/O
  - Not always top consideration
    - Core design and user
  - Skills/experiences even more uncommon
- Resource Management

What is BIG IRON in 200X?

- Large SMPs
  - various levels of NUMA
- Large numbers of inter-connected SMPs

- Isn't MPI the right model for most applications for both types of machine?
  - Increase momentum and reuse of experiences and build SKILLS!!!

Why MPI?

- Data locality is 100% visible
- I/O is either
  - Very local
  - Or non-local
- Programmers commonly find they can minimize need for global state - forced by architecture limitations
- Avoiding sequential thinking becomes natural
  - Not fork and join of OpenMP

- A well designed inter-connect provides for large numbers of simultaneous communications

COMPAQ

Cluster Architecture offers Flexibility

- Flexibility offers potential increased:
  - Capability
- And……
  - Complexity
  - Overhead
  - Reduced efficiency
- But……
  - Ability to tailor machine to a specific mission
  - Imbalance can be good (fewer processors–more interconnect)
- Keeping things simple is almost always better
  - Modularize/separate – partition!
  - Be realistic and specific as possible in defining requirements

COMPAQ

Dimensions of machine design

- Base machine design – vendor + suppliers
- Systems engineering – vendor + customer
  - RFP, Contract, SOW
    - Multiple architects
  - Customer's evolving needs
  - On-site integration details
- Really building an entire environment

COMPAQ

System is as good as its weakest link – availability and performance

- Software (after hardware has stabilized)
  - Scale
  - Complexity
    - Sheer number of facilities, versions, requirements
  - Integration is almost always unique
    - Legacy environment
    - Diverse needs and limited market solutions
    - CONSUMERS need to help Keep It Simple
  - Poor UIs, documentation, repetitive nature
  - Always need to move to newer version of something

Failures will happen – Prioritize

- Scale alone causes a high failure rate
- Prioritize impact minimization
  - Work lost – lost data
  - User job completes to next checkpoint
    - And restart
  - Performance (e.g., RAID configuration)
  - System administration and service effort

Resource Management

- Key to getting the MOST!!
- Core is supplied by vendor
  - Performance, job capture etc…
- Customer requirements vary:
  - LSF
  - PBS
  - Home grown
- Adequate focus to insure high quality integration, scheduling, **failure handling**, etc….?

Practical hardware tradeoffs

- Costs and customer budgets drive compromises
  - RAID 0+1 verses RAID5
  - Standard CPUs/systems verses specialized high availability designs
  - Hot sparing level (nodes/disk etc….)

Core platform design choices

- SMP building block
- Processor
- Memory
- Inter-connect
- Core system software base – not enough time to do justice to today….

SMP building block

- Volume high enough to justify significant enhancements beyond COTS core?
  - Issue of both market size and product life cycle
- Density/cooling
- I/O bus/bridge
  - Where in SMP hierarchy
  - Count – not just slots
- Reliability – memory failures

Processor

- Pin count and protocol to utilize available main memory bandwidth
- Many outstanding loads and stores to memory
  - Compiler support!!
- Compare/focus on sustained performance not peak
- Volume&dollars = influence and innovation

Memory

- Capacity is not an issue
  - Opportunity to exploit replication and mitigate remote latency
    - Cost
    - Programming model
- Reliability become more of an issue with increased capacity
  - Move from ECC to RAIDed memory (EV7)

Memory Hierarchy – hide it, support it

- Computation
  - Page coloring
  - Large pages – efficient TLB usage
  - Core instructions and OS support to actively manage cache/memory flows
  - Compiler cache/memory flow management
- Inter-connect interaction/interference
- I/O interaction/interference

Inter-connect

- Reliability – failure containment!
- Overhead and resource costs
- Latency
- Remote read to support pseudo-shared memory via MPI One-sided/ SHMEM operations
- Bandwidth
- Versatility to support programming clichés
  - Matched sends and receives
  - Unmatched or asynchronous transfers

COMPAQ

Communications assist and programming model

- MPI in some ways is too rich
  - send, bsend, isend, wait, probe
- Vendors strive to allow bad programs to run
  - Need to improve “suggested” usage documentation
- Last machine used and portability issues limit user incentive to optimize
- User’s don’t use tools to understand the fine level details of their programs run time behavior
- Programs are not designed for performance analysis

COMPAQ

Connection - I/O port to memory

- PCI – okay latency
  - 133 (PCI-X) 1 GB/s
  - 266 (DDR) 2 GB/s
  - 512 (QDR) 4 GB/s

- 3GIO – 250 MB/s/lane, latency is TBD
  - 8 lanes 2 GB/s
  - 16 lanes 4 GB/s
  - 32 lanes 8 GB/s

COMPAQ

New standard technologies

- Drive latency and overhead requirements!!!!!!
- Infiniband
- iWarp/RDMA
- xx Gb Ethernet
- 3GIO

Multi-issue and parallelism beyond the CPU

- Inter-connect
  - Within NIC – pipelined, strive to make I/O port the occupancy bottleneck
  - Multiple rails
    - scale bandwidth
    - reduce effective latency
      - Opportunity for use of older or thinned network if locality or bandwidth requirements can be met
    - availability

Multi-issue and parallelism beyond the CPU

- I/O
  - Multiple HBA/paths
    - scale bandwidth
    - Improve request rate to devices
    - Availability – No single point of failure

I/O model and implementation

- Scaleable Parallel design
- High availability
- No single point of failure in hardware
- No bottlenecks end to end

Co-operation and acceptance is required

- To yield the highest effective usage
  - Machine must have **adequate** resources, facilities and capabilities
  - Users must embrace the **limitations** and utilize the machine in a manner that is resilient

Summary

- Distribute machine investment $$ properly
- Build momentum in key areas with focus
  - Tools, resource managers
- Latency improvements will not match bandwidth improvements
  - Vendors must strive to hide latency
  - Users must design programs accordingly
- Increase and improve developer and user skills
- Focus on keeping things simple – the core system is complex enough